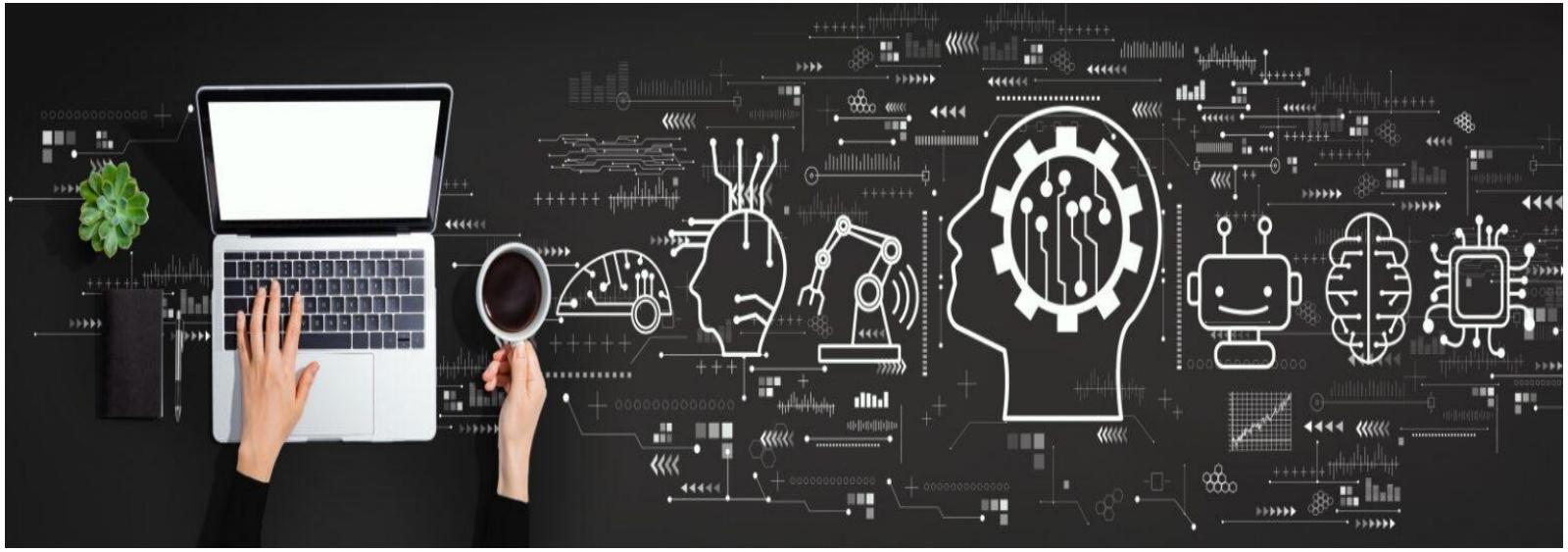


Machine Learning

Course Instructor : Dr. Sarwar Shah Khan

Stacking

Lecture # 05

Stacking

- Stacking (Stacked Generalization) is an **ensemble learning technique** where multiple machine learning models (base learners) are combined to improve predictive performance.
- Instead of using simple averaging or voting, stacking involves training a **meta-learner** (also called a blender) that learns how to best combine the predictions from base models.

Stacking

- Stacking vs. Bagging and Boosting
  - Bagging and Boosting are homogeneous week learners. While Stacking is heterogeneous week learners.

Feature	Bagging	Boosting	Stacking
Training	**Parallel**	**Sequential**	**Parallel + Meta-model**
Focus	Multiple **independent** models	Focuses on **errors**	Learns **from multiple models**
Final Prediction	Majority voting / averaging	Weighted combination	Meta-model prediction
Example	Random Forest	XGBoost, AdaBoost	Combining SVM, NN, Trees

Stacking Working Mechanism

- **Base Learners (Level-0 Models):**
  - Multiple models (e.g., Decision Tree, SVM, Neural Network) are trained on the dataset.
  - Each model makes a prediction on the validation set.
- **Meta Learner (Level-1 Model):**
  - Takes the predictions of the base learners as input features.
  - Learns how to combine the predictions to improve accuracy.
  - Usually, a simple model like Logistic Regression, Random Forest, or XGBoost is used.

Stacking Example

- **Train Base Models (Level-0)**
  - Train Decision Tree, Random Forest, and SVM on the dataset.
  - Get their predictions.
- **Train Meta Learner (Level-1)**
  - Use predictions from base models as features.
  - Train a Logistic Regression model (or any other model) on these features.

List of Stacking Algorithms

- **Simple Stacking**
  - Uses different base models and a meta-learner for final prediction.
- **Weighted Stacking**
  - Assigns different weights to base models based on their performance.
- **Blending**
  - A variation of stacking where a validation set is used for training the meta-model.
- **Stacked Autoencoders**
  - Uses deep learning-based autoencoders to stack multiple feature representations.
- **Multi-layer Stacking**
  - Extends stacking by using multiple layers of base learners and meta-learners.

Four Key Components of a ML Algorithm

- **Hypothesis**
- A hypothesis is a mathematical function that represents the relationship between input features (X) and output (Y).
- It is the model that a machine learning algorithm builds to make predictions.

Example:

- In **Linear Regression**, the hypothesis function is:

$$h(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

where:

- $x_1, x_2, \ldots, x_n$ are input features
- $w_0, w_1, \ldots, w_n$ are model parameters (weights)

- In **Logistic Regression**, the hypothesis function is:

$$h(x) = \frac{1}{1 + e^{-(w_0+w_1x_1+w_2x_2+\ldots+w_nx_n)}}$$

This function outputs values between **0 and 1**, making it suitable for classification problems.

Four Key Components of a ML Algorithm

- **Application**
  - Linear Regression → House Price Prediction
  - Logistic Regression → Spam Email Classification
  - Etc…

Four Key Components of a ML Algorithm

- **Loss Function**
- A loss function measures how well or poorly the hypothesis function predicts the actual target values.
- It calculates the difference between the predicted and actual values.

Example:

- **For Regression:** Mean Squared Error (MSE)

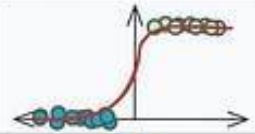
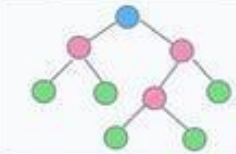
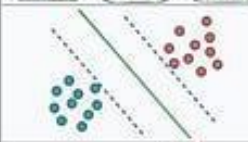
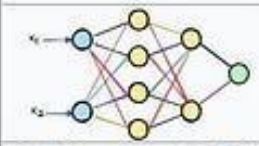
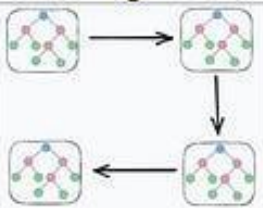
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where y_i is the actual value and $\hat{y}_i$ is the predicted value.

- **For Classification:** Cross-Entropy Loss

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

- This function ensures that if the predicted probability is far from the actual label (0 or 1), the loss is high.

	Linear Regression	Mean Squared Error
	Logistic Regression	Cross-Entropy Loss
	Decision Tree Classifier	Information Gain or Gini impurity
	Decision Tree Regressor	Mean Squared Error
	Random Forest Classifier	Information Gain or Gini impurity
	Random Forest Regressor	Mean Squared Error
	Support Vector Machines (SVMs)	Hinge Loss
	k-Nearest Neighbors	No loss function
$P(B\|A) = \frac{P(\mathrm{B} \cap A)}{P(A)}$	Naive Bayes	No loss function
	Neural Networks	Regression: Mean Squared Error Classification: Cross-Entropy Loss
	AdaBoost	Exponential loss
	Gradient Boosting | LightGBM | CatBoost | XGBoost	Regression: Mean Squared Error Classification: Cross-Entropy Loss

Four Key Components of a ML Algorithm

- **Application:**
  - MSE → Used in predicting stock prices
  - Cross-Entropy Loss → Used in Image Classification
  - Etc…

Four Key Components of a ML Algorithm

- **Derivatives**
- Derivatives measure how much the loss function changes concerning the model parameters (weights). They help determine the direction and magnitude of updates needed to minimize the loss.

Example:

- The derivative (gradient) of the Mean Squared Error (MSE) loss function with respect to weight w is:

$$\frac{d}{dw}J(w) = -\frac{2}{N}\sum(y_i - \hat{y}_i)x_i$$

This tells us how much we should change w to reduce the error.

Application:

- Gradients are used in **backpropagation** in neural networks to adjust weights efficiently.

Four Key Components of a ML Algorithm

- **Optimization algorithms**
- Optimization algorithms adjust the model parameters (weights) based on gradients to minimize the loss function.

Example:

- **Gradient Descent**: Updates weights using the formula

$$w = w - \alpha \frac{d}{dw} J(w)$$

where α is the learning rate.

- **Adam Optimizer**: A more advanced optimization technique that adapts the learning rate for each parameter separately.

Application:

- **Gradient Descent** → Used in training Linear & Logistic Regression models
- **Adam Optimizer** → Used in training deep learning models like CNNs & RNNs

Four Key Components of a ML Algorithm

- Common optimization algorithms
  - Gradient Descent (GD)
  - Stochastic Gradient Descent (SGD)
  - Adam Optimizer